

End-to-End Information Infrastructure & Services
in Open Science Grid (OSG)
Architecture Options

Table of Contents

Definitions & Scope	1
Reasons for Change	1
Service Functionality to be Preserved	2
Infrastructure – OGSA-DAI	2
Infrastructure – MDS4	3
Infrastructure – XAware	3
Infrastructure - OSG Custom	4
Infrastructure – No Change	5
Why not Condor Classads?	5
Evaluation Criteria	6
Reliability	6
Scalability	6
Flexibility	6
Usability	6
Security	6
Development Requirements	7
Maintenance Requirements	7
Comparison	7
Operational Comparison	7
Reliability	7
Scalability	7
Usability	8
Security	8
Development Comparison	8
Flexibility	8
Development Requirements	9
XAware.....	9
MDS.....	9
Maintenance Requirements	9
XAware.....	9
MDS.....	9
References	9

Definitions & Scope

In the Open Science Grid (OSG)[1] there are different types of information that are collected and published. Monitoring information is information that gives a view on resource health and current availability. An example of this would be OSG's Resource and Service Validation (RSV)[2] service. Accounting information gives a view of what resources are being or has been used, for how long, and by whom. The example service for this information Gratia[3]. Service Discovery information provides a view of what services are offered and what those services look like. OSG provides two services that allow an entity to query for services. The Resources Selection Service (ReSS)[4] uses Condor[5] classads to provide service discovery information. ReSS is hosted and administered by Fermilab. The OSG Grid Operations Center (GOC)[6] maintains a BDII service that provides service discovery information in LDIF format to be interoperable with the WLCG[7]. This paper will focus exclusively on service discovery information, its format, and the information delivery architecture. One minor point to note is that the WLCG has a project (Installed Capacity) that redefines some of the pieces of information for pseudo-accounting purposes. This paper will, for the most part, ignore the redefinition of information.

Reasons for Change

The Glue 2.0 service discovery information schema has been officially released and accepted by EGEE[8]. According to EGEE, in order to implement the changes required by Glue 2.0, significant portions of their infrastructure will have to be entirely re-written. The OSG infrastructure is in a little better shape. Portions of the infrastructure need to be worked on, but entire re-writes are not necessary. However, the OSG currently has a somewhat ad-hoc information system that has some inherent fragility built in. An analysis of the current structure is covered elsewhere. With a major change coming, now is a good time to look at whether a new architecture is warranted.

Is the current structure adequate to meet future requirements? Currently, the change in schema for the service discovery information is going to cause a significant amount of development work on core pieces of the infrastructure. This indicates that as time progresses, more changes will require increasing amounts of work on core pieces of the infrastructure. The current infrastructure is probably not flexible enough to make these kinds of changes easily.

Does OSG want to integrate with more grids (OSG is already interoperable with the WLCG)? That may or may not be a goal pursued by the OSG. However, if the OSG decides to pursue interoperability with another grid (whether it is a computing grid like OSG or WLCG, or a data grid like ReddiNet), it will be faced with the potential for adding yet another service like ReSS or the BDII to the mix. The current infrastructure allows for this but with great effort.

Is the current system adequate in terms of reliability and scalability? Testing is currently planned or being conducted. The results of the tests will help determine the answer to this question.

Does OSG want to move to a more universal infrastructure? This paper assumes the answer to this question is yes and provides some suggestions and evaluations of frameworks to achieve a universal infrastructure.

Service Functionality to be Preserved

OSG has a set of services that provide critical functionality. Any new infrastructure must provide a BDII feed to the WLCG information infrastructure. This includes the persistent configuration data that OIM provides.

It also must provide a feed into ReSS for VO's and WMS's that utilize the Condor classads. It will be a while before Glue 2.0 is adopted and implemented by the entire OSG. While the transition is occurring, Glue 1.3 compatibility must be maintained.

FermiGrid[9] makes extensive use of custom attributes to perform matchmaking operations. OSG has a filter in place to remove non-Glue attributes from the feed to the BDII. This functionality must be maintained as well.

Infrastructure – OGSA-DAI

“OGSA-DAI is a middleware product that allows data resources, such as relational or XML databases, to be accessed via web services. An OGSA-DAI web service allows data to be queried, updated, transformed and delivered. OGSA-DAI web services can be used to provide web services that offer data integration services to clients.

OGSA-DAI web services can be deployed within a Grid environment. OGSA-DAI thereby provides a means for users to Grid-enable their data resources.” - <http://www.ogsadai.org.uk/about/ogsa-dai/>

According to the explanation above, OGSA-DAI[10] exposes a database to web services. Currently, the Generic Information Provider (GIP) service does not query any database. All information is gathered via the command line tools provided by the various VDT services and batch managers. OGSA-DAI will add an external software layer that will have to be customized to handle the GIP data (whatever format it is eventually put in). In any case, much of the infrastructure required for OSG would still need to be developed.

Within the OGSA-DAI infrastructure, a proxy server queries the resource in its native format, uses work flows to convert to XML, and caches the data as XML. XML queries can be performed via XPath or Xquery.

A cursory examination of its documentation reveals a comprehensive set of documents. Most of the organizations that published success stories for OGSA-DAI cited the documentation as a key to their success. OGSA-Dai supports SQL, XPath, and file searches. According to the success stories and documentation, it is modular and extensible. Hopefully, this would allow for plugin development as needed.

The LEAD success story[10] for OGSA-DAI highlighted scalability issues in earlier versions. The problem was that the data services were created on a per-request basis. However, in the later versions this problem appears to be addressed.

Infrastructure – MDS4

MDS is the Globus[11] infrastructure for service monitoring and discovery. MDS4, the latest version, uses web services XML messaging to perform queries. To use MDS4, OSG would have to upgrade the version of Globus that it packages. MDS4 has an Aggregator Framework designed specifically for aggregating resources. It includes a Trigger and Index service. Triggers can perform actions based on incoming queries and information. The Index service provides an interface to get resource information. The Framework includes hooks for custom code, allowing for the development of plugins for non-Globus services.

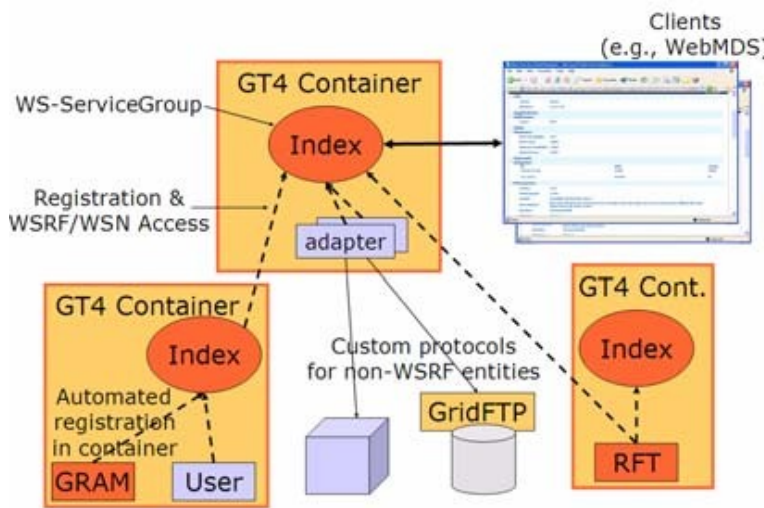


Illustration 1: MDS4 Architecture

A positive to using MDS4 is that The knowledge base in the US is considerable. There is some experience within OSG using Globus tools already and

TeraGrid[12] makes extensive use of Globus and MDS. OSG also has a relationship with Globus.

In an address to the Globus community, Ian Foster made the point that further development and support for MDS as a service registry is a possible goal for 2009. If OSG were to adopt MDS4 for service registry, there may be further support available.

That being said, the documentation for Globus in general and MDS in particular is rather poor. Just searching for an overview of the services and how they fit in an architecture was an arduous adventure.

Infrastructure – XAware

XAware[13] is a data transformation product. It has the ability to query nearly any type of datasource and transform the results into another format. The biggest advantage for this technology is that it provides a single interface for all services. [Illustration 2](#) shows a possible use case for XAware. Here,

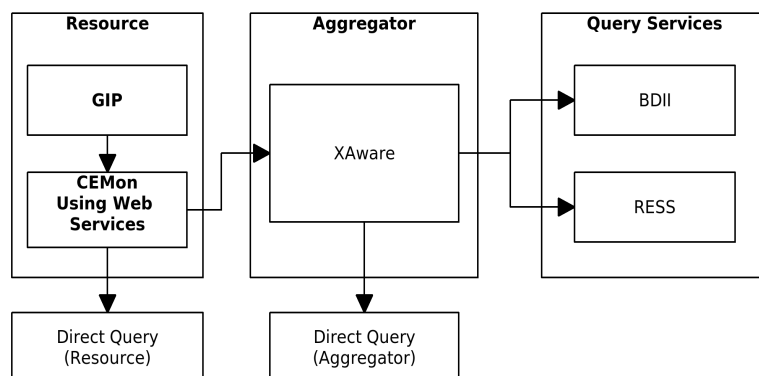


Illustration 2: XAware Architecture

GIP output would be changed to XML rather than LDIF. CEMon would be configured to use the web services that are already built in. Now instead of CEMon *pushing* the GIP information to the collectors, XAware will *pull* GIP information from CEMon. The BDII and ReSS Collectors will need to be modified to ask for information from XAware. XAware will transform the XML that CEMon provides into a format that the

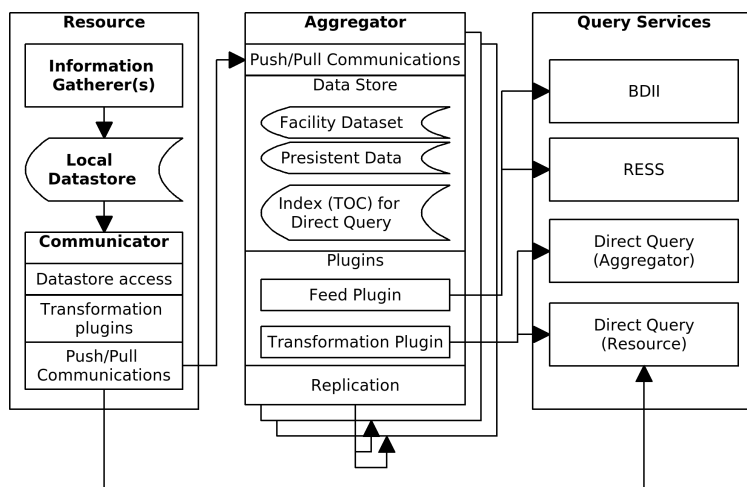
Collectors want to read.

Another advantage to using this architecture is that each piece can be independently queried to so what it is producing and passing along. For example, if the GIP team needs to debug a potential GIP problem, they can directly query the CEMon that is running on the resource to get results for analysis. XAware can be queried directly to get its output as well.

If another form of Query Service is identified and required, then “all” that needs to be done is add another interface within XAware to transform the results into the desired format.

Infrastructure - OSG Custom

A custom solution is always an option. This solution would probably require the greatest amount of development effort. There are also several approaches that could be taken.



OSG could use a similar model to XAware where there is a middle-man service that transforms the GIP results into a usable format for ReSS or BDII. (See [Illustration 2](#))

OSG could also use the MDS model where there are hooks into every service that a resource provides. [Illustration 3](#) gives a view as to how this would potentially be accomplished.

Illustration 3: Custom Architecture

The Resource box can be a Compute Element, a Storage Element, or any service. The Resource has one to many Information Gatherers. As their name implies, the job of the Information Gatherer is to query the resource to gather information about this particular resource. The gatherer will put the information into some form of local datastore. The implementation of that datastore could be as simple as a flat file, similar to CEMon's implementation. The resource will have a communication component running on it (referred to as the Communicator [Illustration 1](#)). The job of the Communicator is to query the local datastore, transform the data into a desired format (if needed) and either initiate push request or accept a pull request and transfer the information to the Aggregator.

The Aggregator box has the job of receiving or pulling service information from the resources and aggregating the information for a logical grouping of resources. There are components for querying the information and transforming the information into other formats.

The Aggregator in [Illustration 3](#) shows three different types of information cached in a datastore. The datastore labeled “Facility Dataset” is intended to show the use case where the Aggregator caches information for all resources within a facility. All information about resources at a facility are aggregated and reported via BDII or ReSS. The “persistent data” datastore represents data that doesn't change often but must be available. An example of this is the information supplied by the OSG Information Management System (OIM) to the WLCG. The last datastore shown is an example of the type of information that would be hosted if the OSG were to support a “peer-to-peer like” model. In this model the Aggregator would become a “super peer”. Each of these datastores would have their own plugins for data management and transformation.

The Query Services query the Aggregator for information. Query Services can be existing services such as ReSS or BDII, or it can be direct queries from WMS's or individual entities. Plugins need to exist on the Aggregator to format information results according to the needs of the requesting Query Service.

Ideally, every component within this architecture can be distributed and/or replicated. It should also be noted that not all the components are required depending on how the final architecture is implemented. The generic architecture is flexible enough to handle a peer-to-peer type network or a static centralized network.

Infrastructure – No Change

Another option to be considered is to leave things as-is. The following issues need to be understood and accepted for this option to be practical.

1. Any changes to the schema versions will result in significant development effort.
2. The potential for a break-down increases over time as changes are made to how information is gathered or exposed.
3. A new query format (other than ReSS or BDII) would require a whole new development project.

Why not Condor Classads?

To be technical, classads are a data storage format not an infrastructure. ReSS would be the infrastructure of choice for using classads. The problem with classads is that there is no structure or schema. While some may view this as a positive, it really becomes a negative when integration with more than a condor match maker is desired. For instance, while possible, it will be very difficult to transform classads to Idif that conforms to the Glue schema. In order to do this, a schema of some type must be imposed on the classads in order to guarantee good, valid, well formed data coming out of the transformation. Since the requirement to have an imposed schema of some type exists regardless of the implementation, it is better to treat ReSS and the BDII as consumers of the information and transform the data into the required formats.

While ReSS is good for high performance matchmaking, the infrastructure to populate ReSS is

somewhat fragile (a minor schema-valid change to the information reported can cause an unmanageable number of classads to be generated) and one component (CEMon) is prone to cryptic failure modes.

Evaluation Criteria

As with any product or service a set of criteria must be established to properly evaluate and compare similar items.

Reliability

Reliability is a measurement of data integrity and availability. Available services are services that are running and responding to requests. Availability is defined as a percentage of time that services are available. For example, FermiGrid desires to have its HA branded service availability to be greater than 99.999%. Data integrity is equally important. It does no good to have a service available 100% of the time but returning garbage. For as long as the infrastructure is considered available, the information flow into and out of the infrastructure must be complete and accurate.

Scalability

A measurement of scalability is responsiveness. How many queries per some time period can the infrastructure handle? What is the upper limit on the quantity of distinct sets of information received, stored, and reported by the infrastructure? What is an acceptable response time between issuing a query and receiving a result? What options are there to mitigate bottlenecks?

Flexibility

One of the reasons for change is the new schema. The current architecture is, arguably, not flexible enough to handle this change, prompting this paper. Any new infrastructure must be architected such that changes to a schema can be handled without breaking operations with the previous schema. It should be able to export different views of data (ldif, classad, xml, etc) as well. It would be helpful to have the new infrastructure handle multiple query languages (SQL, XPath, ldapsearch, etc). Basically, the question being asked is, "How difficult is it to make changes"?

Usability

Usability can be a loaded word with different meanings for different people. For this evaluation, usability refers to a well documented infrastructure that has mechanisms to log and return meaningful results. These results include requested information and any information about error conditions. One requirement that should be strongly pushed for is that any new infrastructure must have meaningful logging that is easily accessible by administrators.

Security

In reality, security should have its own evaluation rubric. However, here it will refer to identification and authorization of entities that will interact with the infrastructure at all levels.

Development Requirements

Development requirements refers to the amount of development required to implement an infrastructure. What components need to be written? How much is provided out of the box? What development is required to “glue” all the infrastructure components together? What development effort will be required to provide operational support (bug fixes, new requests, validation software)?

Maintenance Requirements

Maintenance refers to administering the infrastructure. What is the learning curve for the various administrators of the components within the infrastructure? What will the upgrade path be?

Comparison

As discussed above OGSA-DAI wasn't really designed for service discovery. Therefore, that piece of technology will not be discussed here. Condor classads are not really an infrastructure, so they won't be discussed either. Last, the assumption in this section is that a change will be made. Therefore, the no-change option will not be discussed. The two options that will be discussed are MDS4 and XAware. It will be assumed that the custom solution will be developed to fit the criteria of the evaluation.

Operational Comparison

Reliability

MDS4 appears to be reliable in terms of availability and integrity. TeraGrid and caBIG both use MDS4 as their service registry and seem satisfied with it.

XAware is being used for some software development internally at Fermilab. The developers who use the software appear to like it very much. Members of the team who also provide operational support to ReSS also appear to have a favorable opinion of it.

Scalability

MDS4 utilizes the Java WSRF framework. In an address to the Globus community Ian Foster mentioned that WSRF is a potential performance and scalability bottleneck. Globus is in the process of evaluating different technologies to replace WSRF, so for the short term, OSG would have to accept any scalability issues with MDS if that is the technology of choice.

XAware would make use of existing components within the OSG software stack. The two scalability issues would be how often can XAware be queried before a service interruption occurs and how often XAware can query a resource before the resource experiences interruptions. These are currently unknown quantities, however in the case of an overload of XAware, it is possible to have an instance of XAware run for each Query Service.

Usability

XAware has central logs that contain any debugging information. The developers at Fermilab have not seen any XAware internal errors yet so they do not know how those errors get logged. The process errors are defined by the developer. Those errors are both logged in the XAware logs and sent to the requesting endpoint. The usability of the error messages are dependent on the developers. So in the end, the requesting endpoint will get an error or the content requested.

MDS is a Globus product. Past experience with Globus products has shown that the logging that occurs has limited usefulness from an operations standpoint. The logs tend to be cluttered with miscellaneous information that appears to have little to no bearing on the problems that appear. The learning to translate the logging messages could be steep. It is unknown what the requesting endpoint sees when an error occurs.

Security

Currently, the information that GIP provides is considered “public”. In other words it is necessary to expose that information to anyone who asks for it. Therefore, we do not need to worry as much with authentication and authorization issues.

In the current OSG model, CEMon “pushes” the GIP information to the ReSS and BDII collectors. MDS4 and XAware both change that model from “push” to “pull”. MDS4 and XAware query resources for information and process that information. In a “pull” model, the reporting resource needs to have a port open to incoming requests. OSG may or may not want to consider having some form of authorization process, like an OSG service certificate that will restrict resource query access to OSG approved services. Doing this may help mitigate against attacks on that port.

There may be other security concerns that the OSG security group might bring up as well.

Development Comparison

Flexibility

As far as flexibility is concerned XAware is extremely flexible. The usage envisioned for OSG is that of a conduit for the GIP Information. To add new query services, one must define the transformation that must take place in XAware. Then the query service can get the information from XAware. No changes are necessary on any of the resources. XAware also has features that allow changes to be versioned, so a query service could ask for a specific version of a transform. This would help insulate the query services from changes that may break one query service but is required by another service.

MDS has deep hooks within Globus so any Globus service is enabled out of the box. However, any non-Globus service needs to have “plugins” written to expose its information in a way that MDS can understand. Some form of feed plugin must be developed for each query service. New query services mean new feed plugins. MDS provides a way to query for services natively as well.

Development Requirements

There will be a learning curve for developers to use either XAware or MDS.

XAware

Developers will need to adjust GIP to output XML. In this case it might be desirable to output in Glue 2.0 compliant XML. XAware transformations will need to be developed to transform the GIP information to Glue 1.3 compliant LDIF, Glue 2.0 LDIF, and Condor Classads. The Collectors for both BDII and ReSS will need to be modified to query XAware for information.

MDS

MDS4 is deeply integrated with all Globus components already so the GIP may be replaced or eliminated in this architecture. Developers will need to write “adapters” to collect and expose information from non-Globus services. Some type of feed plugin will be needed to push/pull information to the BDII and ReSS collectors.

Maintenance Requirements

XAware

XAware versions are released roughly every quarter. Upgrading to these versions is not necessary unless some desired feature or bug fix is included. Documentation is pretty thorough and there seems to be an active community within the XAware forums. Support can also be purchased if desired.

MDS

MDS documentation is minimal and hard to find. Since OSG already has a relationship with Globus there will probably be access to greater support. The release schedules for Globus ended in October 2008 with no further release scheduled on their website. Because of this, the upgrade path and schedule is unknown for MDS.

Bibliography

1. Open Science Grid, 2009, <http://www.opensciencegrid.org/>
2. Open Science Grid: Resource and Service Validation, 2009, <http://rsv.grid.iu.edu/documentation/>
3. OSG Gratia Accounting Services, 2009,
<https://twiki.grid.iu.edu/bin/view/Accounting/InstallationGuideVDT>
4. OSG Resource Selection, 2009, <https://twiki.grid.iu.edu/bin/view/ResourceSelection/WebHome>
5. Condor Project Homepage, 2009, <http://www.cs.wisc.edu/condor/>
6. OSG Grid Operations Center, 2009, <http://www.grid.iu.edu/>
7. Worldwide LHC Computing Grid, 2009, <http://lcg.web.cern.ch/LCG/>
8. Enabling Grids for E-sciencE, 2009, <http://www.eu-egee.org/>
9. FermiGrid, 2009, <http://fermigrd.fnal.gov/>
10. OGSA-DAI: Welcome to OGSA-DAI, 2009, <http://www.ogsadai.org.uk/>
11. GT Information Services: Monitoring & Discovery System (MDS), 2009,
<http://www.globus.org/toolkit/mds/>
12. TeraGrid, 2009, <http://www.teragrid.org/>
13. XAware 2009, <http://www.xaware.org/>